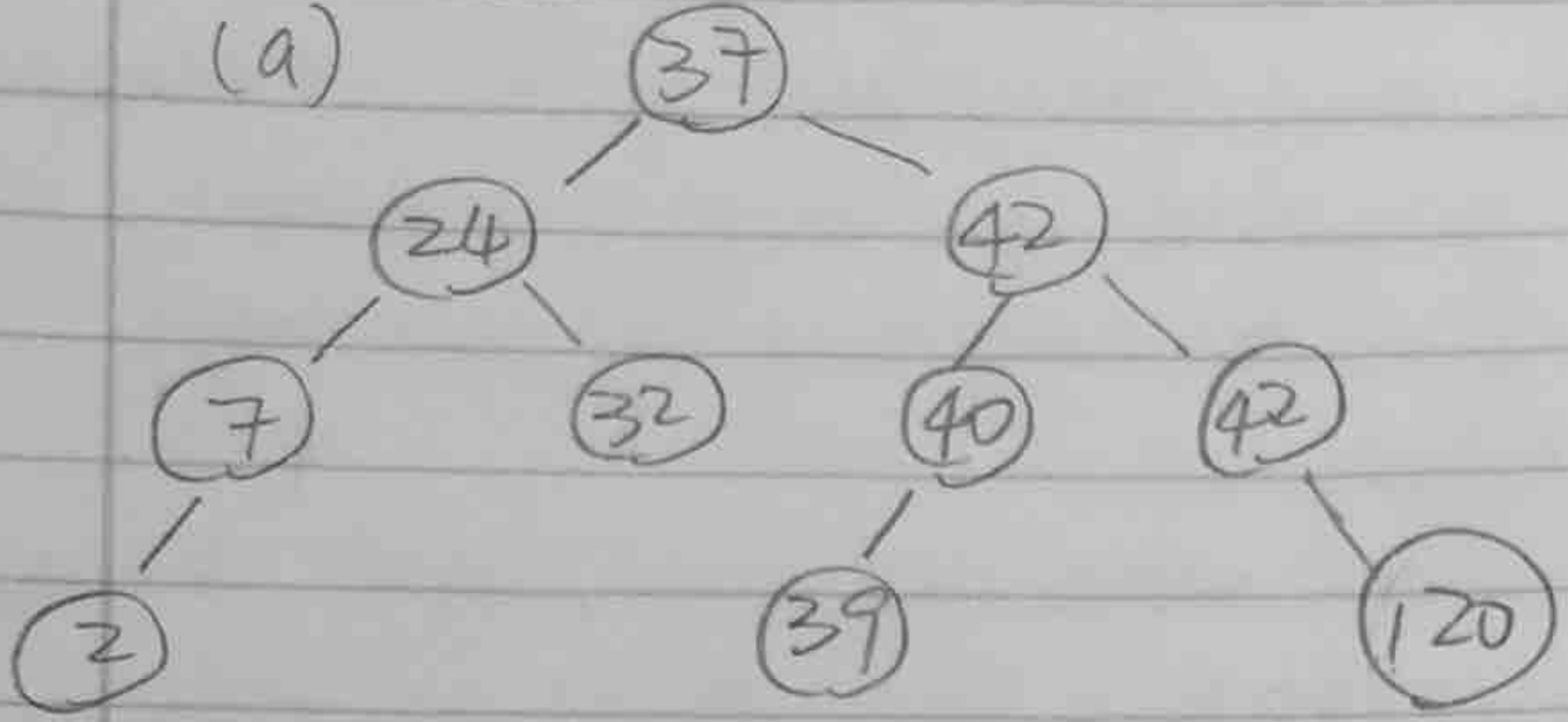
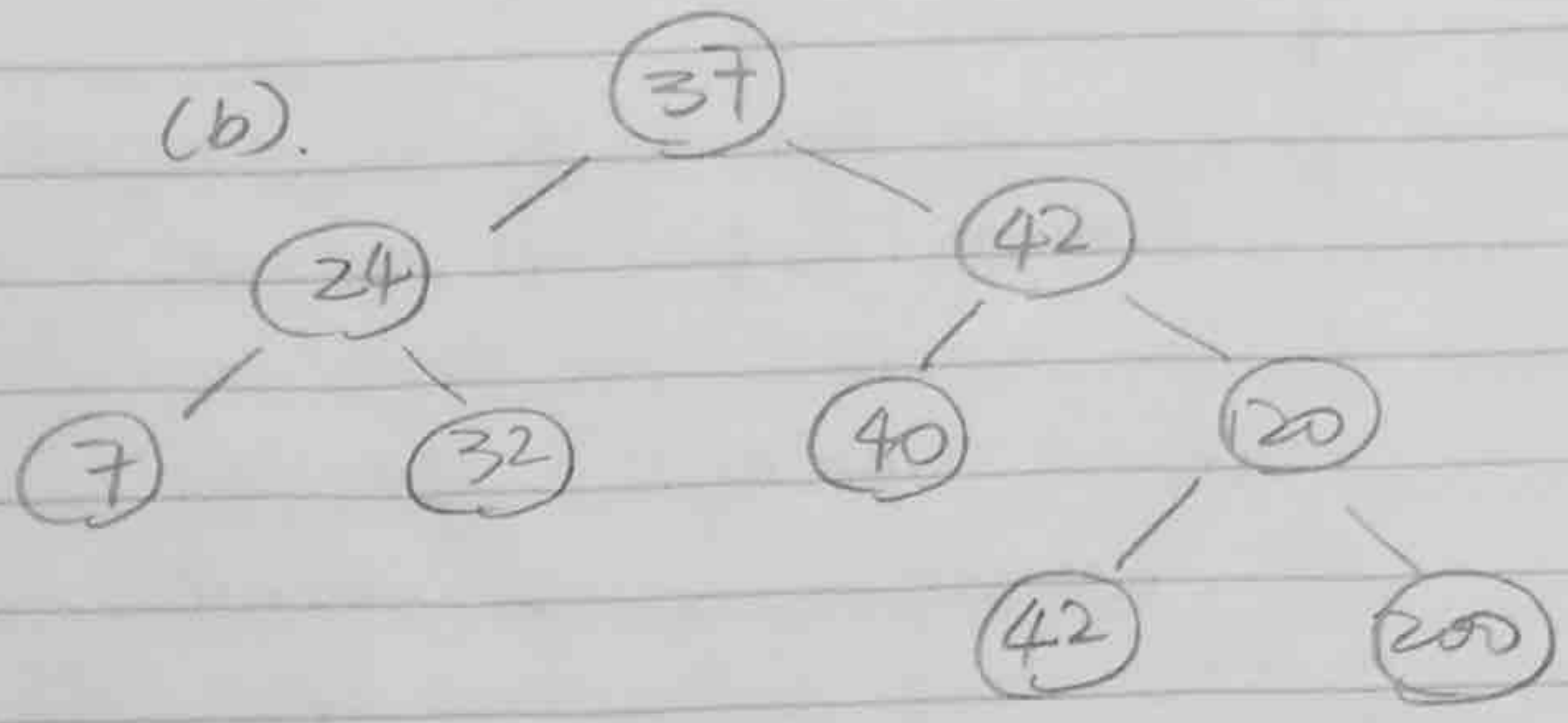
HW 3

Exercise 13.5

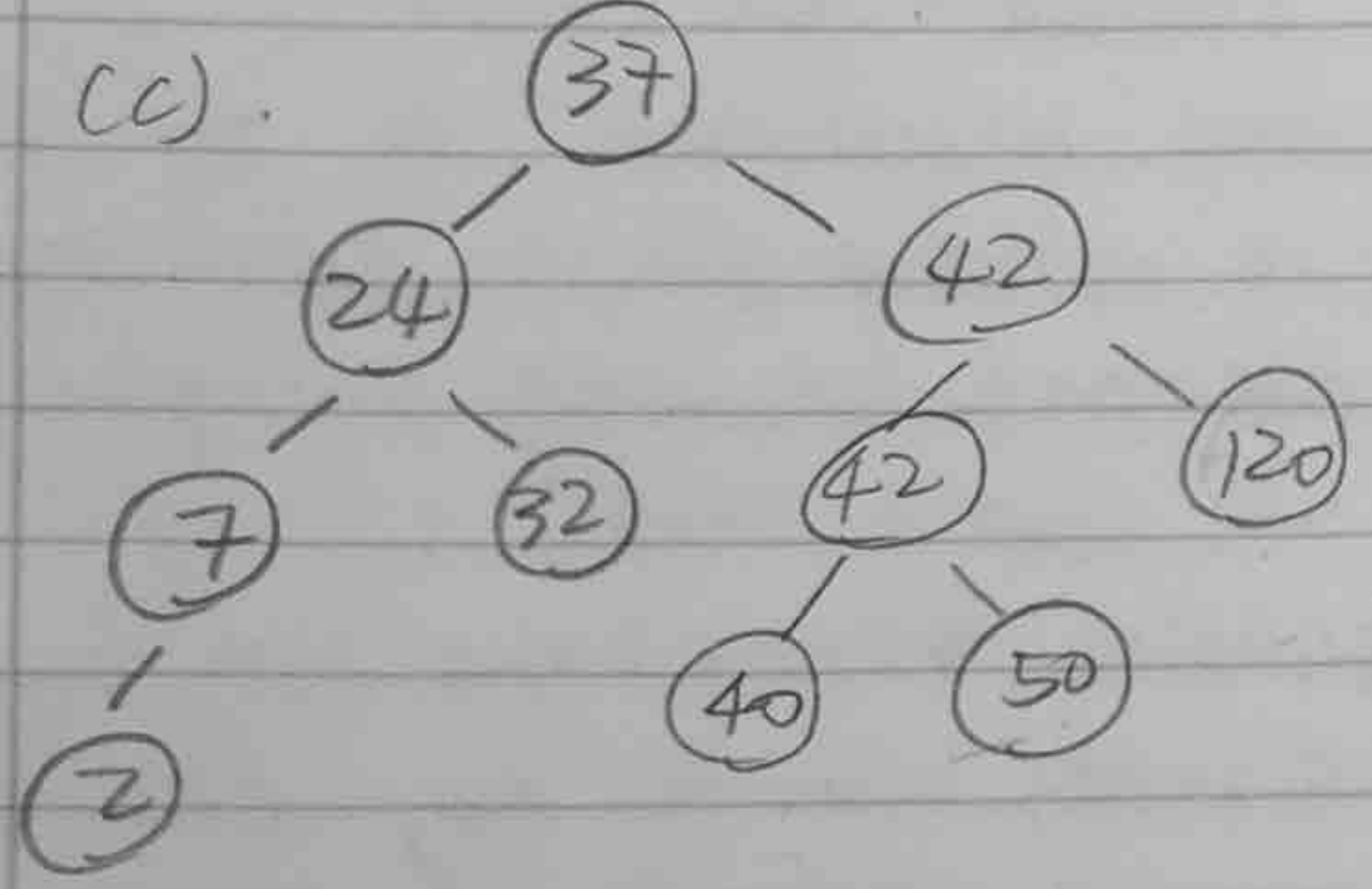
(a)



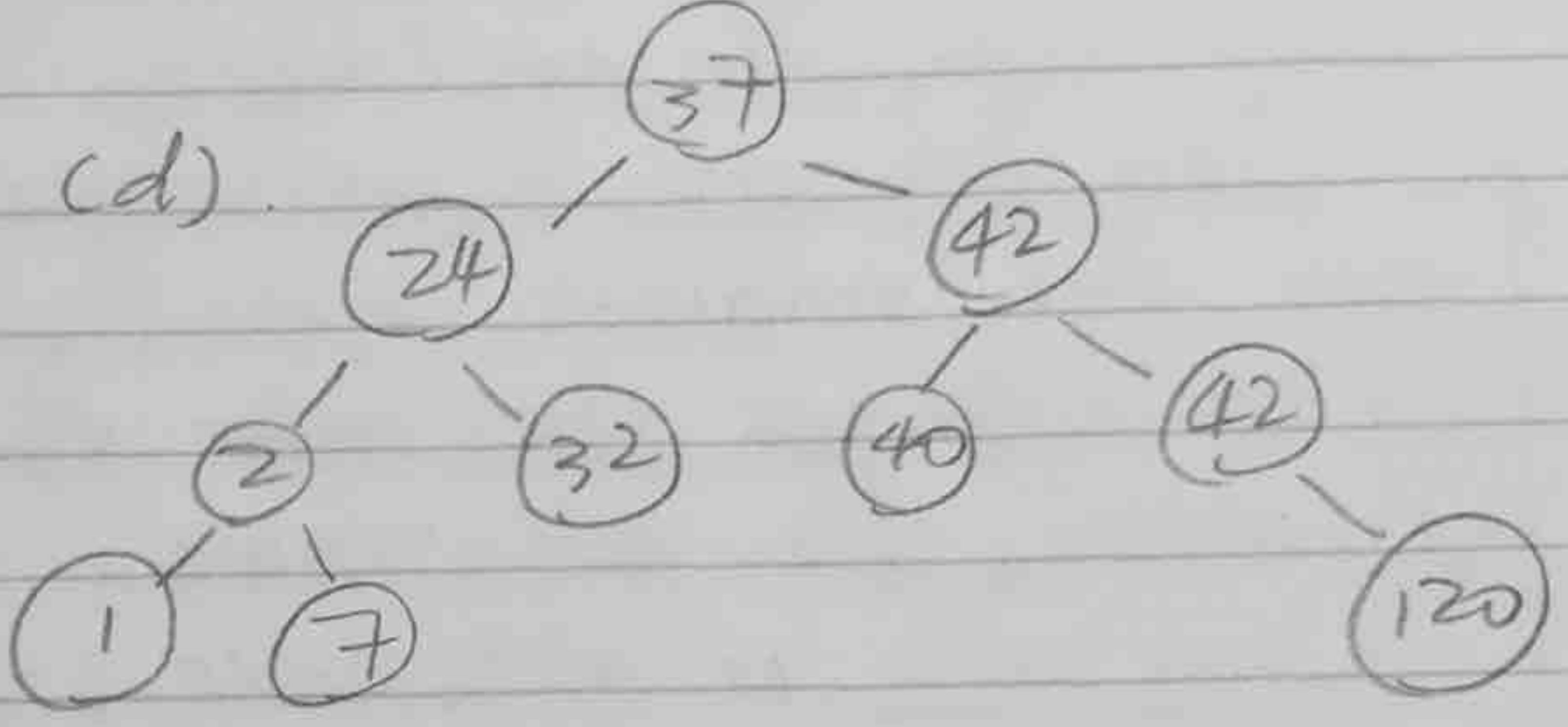
(b).



(c).



(d).



Exercise 7.1

```
Static <E extends Comparable<? super E>>
void sort (E[] A) {  Boolean[] flag;  // add flag variable.
     for (int i = 0 ;  i< A.length-1 ; i++) {
       for (int j = A.length-1 ; j > i ; j--) {
           if ((A. [j]. compareTo( A[j-1]) < 0)) {
               flag[j] = false ;
               Dsutil. swap (A, j, j-1) ;
       } }
     // test
      Boolean temp = true ;
      for (Boolean b : flag) {
         if( b == false) {
              temp = false; } }
      if (temp == true)
           break ;
     } }.
```

Exercise 7.6.

· Insertion sort is stable, b/c when insertion sort iterates through a list of records, each record is inserted in turn at the correct position, so, the order of multiple objects with equal keys don't change. Therefore it's stable.

· Bubble sort is stable, b/c when we compare and swap the two adjacent keys during iteration, the objects with same keys do not move, so it's stable.

· Selection sort is not stable, b/c its algorithm is to find the kth smallest object and swap it to the (k-1)th position, so it changes the order of each element during swapping. Therefore it's not stable.

· Shell sort is not stable, b/c shell sort break the list into sublists, sort them and recombine the sublists. So, it may change the relative order of elements with equal values.

· Mergesort is stable, b/c in merge sort, if there're two objects with the same key, they will not swap each other. So the order of them do not change.

· Quicksort is not stable in efficient implementations, b/c even though quick sort do not swap the duplicates, it may swap other elements beyond the duplicates, so the order of duplicates will no longer be preserved, so it's not stable.

· Heap sort is not stable b/c the operation on the heap can change the relative order of equal objects.

· Bin sort is stable when the sort in each bucket is stable, since bucket sort inserts objects into buckets in order including duplicates, so binsort is stable as long as every bucket is stable.

· Radix sort is stable b/c it sorts data with integer keys by grouping keys by their digits. So it does not change the order of duplicates.

Exercise 7.16.

(a). To sort three numbers, you can find the smallest by comparing two numbers, and swap it into the first position. Then compare the remaining two and swap if neccessary. For the best case, it takes 3 comparisons and 0 swap, for the worst case, it takes 2 swaps and three comparisons, for the average case, it takes 1 swaps and three comparisons.

```
void sort ( int a, int b, int c) {
    swap (a, min(a, min(b, c)));
    Swap ( b, min (b,c)); }
```

(b). To sort five numbers, assume the array is {a, b, c, d, e}

- First group the first 4 nums in Pairs = (a,b), (c,d).
- compare each pair: eg. a<b, c<d.
- Compare the Smallest element in each pair: eg. a<c.
- Compare e with the biggest element b/w a & c
    - if e<c, then compare a,b and e (3 comparisons), finished.
    - if e>c, compare d & e:
        - if d<e, then compare b & d:
            - if b>d, compare b & e, finished.
            - if b<d, compare b&c. finished.
        - if d>e, compare b & e =
            - if b>e, compare b & d, finished.
            - if b<e, Compare b & c, finished.

For the best case, it takes 4 comparisons.
For the worst case, it takes 8 comparisons.
For the average case, it takes 6 comparisons.